Standard Code Library

FLself

SCUT

June 1, 2022

Contents

一切的开始

一些宏定义

- 需要 C++11

```cpp
#include<bits/stdc++.h>
#define int long long
#define PII std::pair<int, int>
#define VI std::vector<int>
#define VPII std::vector<std::pair<int, int> >
#define VVI std::vector<std::vector<int> >
#define ALL(a) (a).begin(), (a).end()
#define SIZ(a) ((int)(a).size())
#define rep(i, l, r) for (int i = (l); i <= (r); ++i)
#define repv(i, a) for (int i = 0; i < (int)(a).size(); ++i)
#define lowbit(x) ((x) & (-(x)))
#define lbpos(x) (__builtin_ctz(x))
#define hbpos(x) (__builtin_clz(x))
template<typename T> std::istream &operator>>(std::istream &is, std::vector<T> &vec) { for (auto &x: vec) is >> x;
↪ return is;}
template<typename T> std::ostream &operator<<(std::ostream &os, const std::vector<T> &vec) { os << '{'; for (auto
↪ &x: vec) os << x << ", "; return os << '}';}
template<class Tuple, std::size_t... Is> void print_tuple_impl(std::ostream &os, const Tuple &t,
↪ std::index_sequence<Is...>) { ((os << (Is == 0? "" : ", ") << std::get<Is>(t)), ...); }
template<class... Args> std::ostream &operator<<(std::ostream &os, const std::tuple<Args...> &t) { os << "(";
↪ print_tuple_impl(os, t, std::index_sequence_for<Args...>{}); return os << ")"; }
struct outputerr {
    inline outputerr &operator<<(std::string &s) { std::cerr << s; return *this; }
    inline outputerr &operator<<(const char *s) { std::cerr << s; return *this; }
    inline outputerr &operator<<(char ch) { std::cerr << ch; return *this; }
    template<typename V> inline outputerr &operator<<(V x) { std::cerr << x; return *this; }
    // template<typename V> inline outputerr &operator<<(std::vector<V> &vec) { std::cerr << "{"; for (auto x: vec)
↪   (*this) << x << ", "; std::cerr << "}"; return *this; }
    template<typename S, typename T> inline outputerr &operator<<(std::pair<S, T> pp) { (*this) << "(" << pp.first <<
↪   ", " << pp.second << ")"; return *this; }
    template<typename V> inline outputerr print(V vec[], int n) { std::cerr << "{"; for (int i = 0; i < n; ++i)
↪   (*this) << vec[i] << ", "; std::cerr << "}\n"; return *this; }
} err;


// -----如果只能终端调试可以换用这个 debug----------------------------------------
#ifdef DEBUG
#define dbg(x...) do { cout << "\033[32;1m" << #x << " -> "; err(x); } while (0)
void err() { cout << "\033[39;0m" << endl; }
template<template<typename...> class T, typename t, typename... A>
void err(T<t> a, A... x) { for (auto v: a) cout << v << ' '; err(x...); }
template<typename T, typename... A>
void err(T a, A... x) { cout << a << ' '; err(x...); }
#else
#define dbg(...)
#endif
// ---------------------------------------------------------------------------
```

数据结构

ST 表

- 二维

```cpp
int f[maxn][maxn][10][10];
inline int highbit(int x) { return 31 - __builtin_clz(x); }
inline int calc(int x, int y, int xx, int yy, int p, int q) {
    return max(
        max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
        max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
    );
}
void init() {
    FOR (x, 0, highbit(n) + 1)
    FOR (y, 0, highbit(m) + 1)
        FOR (i, 0, n - (1 << x) + 1)
        FOR (j, 0, m - (1 << y) + 1) {
            if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
            f[i][j][x][y] = calc(
                i, j,
                i + (1 << x) - 1, j + (1 << y) - 1,
                max(x - 1, 0), max(y - 1, 0)
            );
        }
}
inline int get_max(int x, int y, int xx, int yy) {
    return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
}
```

数学

类欧几里得

- $m = \lfloor \frac{an+b}{c} \rfloor$.
- $f(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \ge c$ or $b \ge c$ 时，$f(a,b,c,n) = (\frac{a}{c})n(n+1)/2 + (\frac{b}{c})(n+1) + f(a \bmod c, b \bmod c, c, n)$；否则 $f(a,b,c,n) = nm - f(c, c-b-1, a, m-1)$。
- $g(a,b,c,n) = \sum_{i=0}^{n} i \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \ge c$ or $b \ge c$ 时，$g(a,b,c,n) = (\frac{a}{c})n(n+1)(2n+1)/6 + (\frac{b}{c})n(n+1)/2 + g(a \bmod c, b \bmod c, c, n)$；否则 $g(a,b,c,n) = \frac{1}{2}(n(n+1)m - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1))$。
- $h(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor^2$: 当 $a \ge c$ or $b \ge c$ 时，$h(a,b,c,n) = (\frac{a}{c})^2 n(n+1)(2n+1)/6 + (\frac{b}{c})^2(n+1) + (\frac{a}{c})(\frac{b}{c})n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2(\frac{a}{c})g(a \bmod c, b \bmod c, c, n) + 2(\frac{b}{c})f(a \bmod c, b \bmod c, c, n)$；否则 $h(a,b,c,n) = nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a,b,c,n)$。

图论

LCA

- 倍增

```
void dfs(int u, int fa) {
    pa[u][0] = fa; dep[u] = dep[fa] + 1;
    FOR (i, 1, SP) pa[u][i] = pa[pa[u][i - 1]][i - 1];
    for (int& v: G[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}

int lca(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);
    int t = dep[u] - dep[v];
    FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
    FORD (i, SP - 1, -1) {
        int uu = pa[u][i], vv = pa[v][i];
        if (uu != vv) { u = uu; v = vv; }
    }
    return u == v ? u : pa[u][0];
}
```

计算几何

二维几何：点与向量

```
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
```

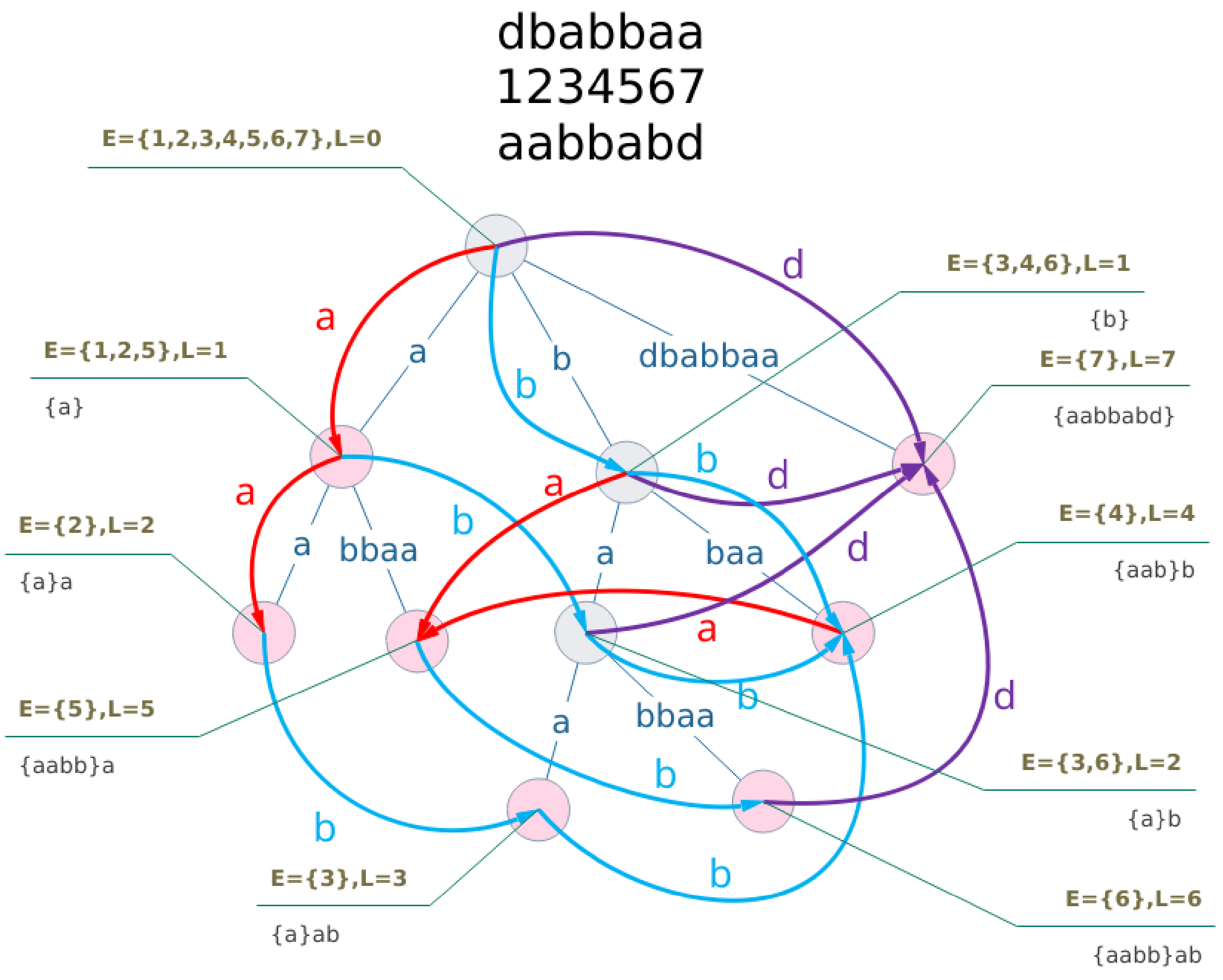
```cpp
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
    explicit P(const L& l);
};
struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
};

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
inline bool operator < (const P& a, const P& b) {
    return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
}
bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
P::P(const L& l) { *this = l.t - l.s; }
ostream &operator << (ostream &os, const P &p) {
    return (os << "(" << p.x << "," << p.y << ")");
}
istream &operator >> (istream &is, P &p) {
    return (is >> p.x >> p.y);
}

LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
// --------------------------------------------
```

字符串

后缀自动机



杂项

STL

- copy

```
template <class InputIterator, class OutputIterator>
  OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result);
```